Web Caching: Locality of References Revisited

Annie P. Foong, Yu-Hen Hu, Dennis M. Heisey
Department of Electrical and Computer Engineering
Department of Surgery
University of Wisconsin
1415 Engineering Drive Madison, WI 53706, USA

Abstract-- An effective caching strategy depends on utilizing locality of reference. Locality is well-researched in traditional caches. However, due to differences between traditional and web caches, such knowledge cannot be applied verbatim. We propose that web accesses exhibit at least four dimensions of locality. A Logistic Regression model is used to confirm our hypothesis. Based on these findings, we implement a web-caching agent to first acquire knowledge about the objects it encounters, and deduce an effective cache strategy dynamically. Results (based on object and byte hit ratios) on trace driven simulations of six web and proxy servers are very encouraging.

Index Terms-- Caching algorithms, logistic regression, locality of reference, multi-dimension, adaptive, web model.

I. INTRODUCTION

Improving the performance of a diverse network such as the World-Wide Web is a daunting challenge. Web caches had been successfully deployed to alleviate problems of congestion and slow response times on the Web [1]. They can be located at clients, servers or proxies. A client cache circumvents the need for any HTTP connection if an object is re-accessed. Similarly, when a group of users share a proxy cache, they retrieve commonly visted pages (e.g. news articles) only once. Finally, a server cache reduces server "think time" by placing high dmand objects on its local hard disk or memory. The savings can be substantial for busy servers. Successful caching ultimately translates to faster response times for end users, lower loads on servers and less congestion on the Internet backbone.

The success of caches can be attributed to the exploitation of the principle of locality. Locality and its effects in traditional caching had been an on-going area of research for many years [2]. Similar research efforts had not been given to web caching. As for commercial products, they achieve higher performance by brute-force addition of hardware and memory. Although this had allowed for rapid development of products, such approaches aim to find quick cures for symptoms, without understanding the underlying issues.

In this paper, we focus on using locality in the prediction of future accesses. Our motivation is two-fold: a) Determine what constitutes web locality and a good method for determining such locality; b) Propose cache strategies based on such locality. A web cache encounters more dimensions of locality than are taken into account by traditional methods. We propose four localities of web references. We shall use a Logistic Regression (LR) model to identify and quantify these localities. The model enables us to predict the re-access probabilities of web objects when given an associated set of features. Objects with high probabilities of re-access will be admitted and possibly pre-fetched into cache. We begin by giving a detailed summary of what other researchers have done in this area, and justifying the directions we take.

II. RELATED WORK

If a priori knowledge is available, the optimal eviction policy for a two-level system cache, dealing with fixed-size pages, is the well-known Longest Forward Distance (LFD) algorithm [3]. More recently, Volker extends this work and deduce a near-optimal algorithm for three-level caches in a global memory system [4]. However, when pages (or objects) are of variable size, as in the case of web caches, the problem becomes NP-hard [5]. When future knowledge is not available, the least-recently-used (LRU) and least-frequently-used (LFU) eviction policies are well accepted for traditional system caches. A similar consensus had not been reached for web caches. A multitude of heuristic-based eviction algorithms exists, each to varying degrees of success. They include variations of the LRU and LFU algorithms, with size and last accessed times as secondary considerations [6, 7, 8]. Recommendations are based mainly on empirical observations. A strategy is first suggested and then tested to determine its performance. Given enough manipulation and fine-tuning, one is able to find an algorithm that will work well for a given web workload.

Wooster infered locality of reference by noting what cache algorithms work best [9]. For example, if LRU performs well for a trace, temporal locality is inferred. Others infer the existence of locality by noting hit ratios in web traces [7, 10]. However, average hit ratios provide only a general measure of usage, not specific locality details. In addition, there is no way of characterizing multiple localities. Moreover, conflicting

---

Foong is currently with the Server Architecture Lab at Intel Corp. This work was done when she was a student at the University of Wisconsin – Madison.

results [7, 11] are common since statically assigned algorithms will not adapt to different sites.

There are also several attempts to deduce a universal model that applies to the entire Web. Lorrenzetti and Rizzo derived a re-access probability as a function of file sizes, number of last access and time since last access [12]. The probability is obtained by manually fitting an exponential function to empirical data. Their observation is for one web workload and the reproducibility of the model in different situations is questionable. Arlitt and Williamson identified ten web invariants (including the distribution of file types, sizes, one-time referencing, etc) which they believed to be characteristics of all web workloads [13]. Cunha et al noted that most Web files are biased towards smaller sizes. They therefore suggested a caching strategy that keeps smaller objects in cache. Such general models provide valuable insights to the overall workings of the Web and form the basis for developing realistic web workloads. However, these observations give typical values over the long run. They may not be indicative of web accesses a cache may see during the period of interest. For example, researchers have found that HTML and image files each account for half of web accesses [9, 13]. We cannot tell from access frequencies which is more cache-worthy. However, our studies did show that HTML files are more likely to be re-accessed than image files. Therefore, for a model to be useful for caching, we need to characterize, not just web *accesses*, but web *re-accesses*.

Based on these observations, we make the following recommendations. Effective web cache strategies are based on more than one feature. Any methodology used must be able to model multiple features adequately. Such a model must also be simple and easily replicable. A Logistic Regression (LR) model is one such method. We will take a reverse route from other researchers. Instead of testing a variety of cache strategies to determine what works best, we use the model to project strategies that work. We want a model that can adapt to the nuances of specific sites. We are not only interested in a model's ability to fit current data, but also in its ability to predict. Our assumption is that the web *re-access*, given a set of *features*, is *predictable*.

III. THE CACHE MAINTENANCE PROBLEM

A cache essentially takes 3 actions: admittance, eviction and pre-fetch (Figure 1). We define the lifespan of an object to be the time between two successive accesses of that object. We have shown in earlier works [14, 15] that an effective cache strategy is dependent on the accuate prediction of lifespans. We refer interested readers to those papers and summarize pertinent points here:

A. Cache admittance:

A web cache is under no obligation to cache an object at least once. This unique web architecture creates a cache admittance strategy, versus a strictly replacement one. Cache admittance requires determining if an object has an infinite lifespan. An object is considered to have an infinite lifespan if its next access is larger than the end of a web trace, its expected expiration (including dynamic objects), or a user-assigned threshold. Any object with an infinite lifespan will not be admitted into cache at all.

B. Cache eviction:

A cache eviction will take place when cache is full. To fit as many objects into a given cache resource, the object with the largest weighted lifespan (size*lifespan) is a good candidate for eviction if optimizing for object hit ratios. If optimizing for byte hit ratios, a good candidate for eviction is the object with the largest lifespan [14].

C. Cache pre-fetch:

We denote the actual requested web objects as primary objects, and links accessible from the primary objects as secondary objects. If an object is likely to be requested in the near future, it should be pre-fetched. For our purposes, only secondary objects of the current request can be pre-fetched. Secondary objects with the smallest weighted lifespans is pre-fetched first. Pre-fetch will continue until the next primary object arrives. Pre-fetch is an extensive topic with many possibilities. Our purpose here is to show how the framework we are developing for admittance and eviction can also be applied to pre-fetch.

Since object lifespan is inversely proportional to its re-access probability, the probability given by the LR model can be used. In the following sections, we shall describe how we make use of locality to make such predictions.

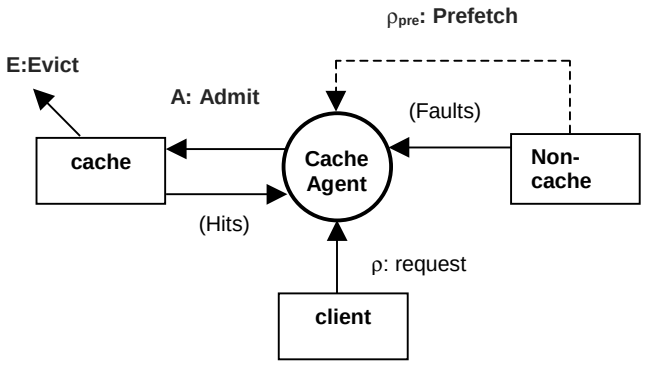

Figure 1 Actions taken by a web cache

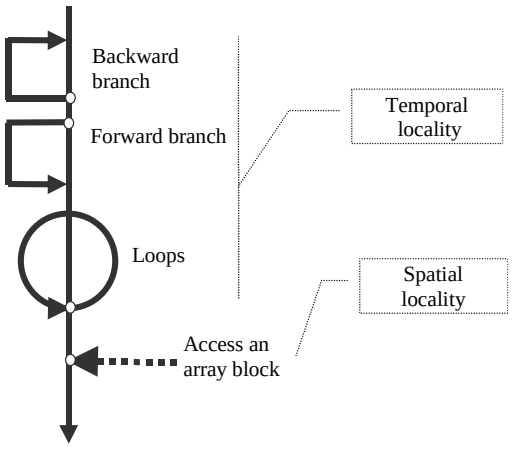

Figure 2 Code traversal

IV. LOCALITY OF REFERENCE IN WEB ACCESSES

Locality is primarily dictated by the traversal methods available. In the case of traditional caches, that locality is generally limited by how code executes. Most programs are of a sequential nature [16], and the result is therefore spatial and temporal locality (Figure 2). Temporal locality forms the basis of LRU; and spatial locality, the basis of pre-fetching.

Caches had not worked as well in distributed databases and vector machines due to the "multi-dimensional" access of such applications. We believe the same to be true of web accesses. Many researchers have found LRU to perform poorly for web caches [6, 7]. LRU makes use of only one dimension in the locality principle. When more dimensions exist, LRU, becomes sub-optimal. To ensure that we make full use of available information, we need to include more dimensions of locality. Extracting multi-dimensional locality can be tricky. We must first determine quantifiable features (predictors) that can adequately describe each type of locality.

For web accesses, the traversal methods available to users (via a browser) are numerous. Most browsers support the following methods:

- Explicit URL
- In-text links
- Back/Forward
- History list
- Bookmark list

As such, we propose that there are at least four dimensions to web cache locality (Figure 3):

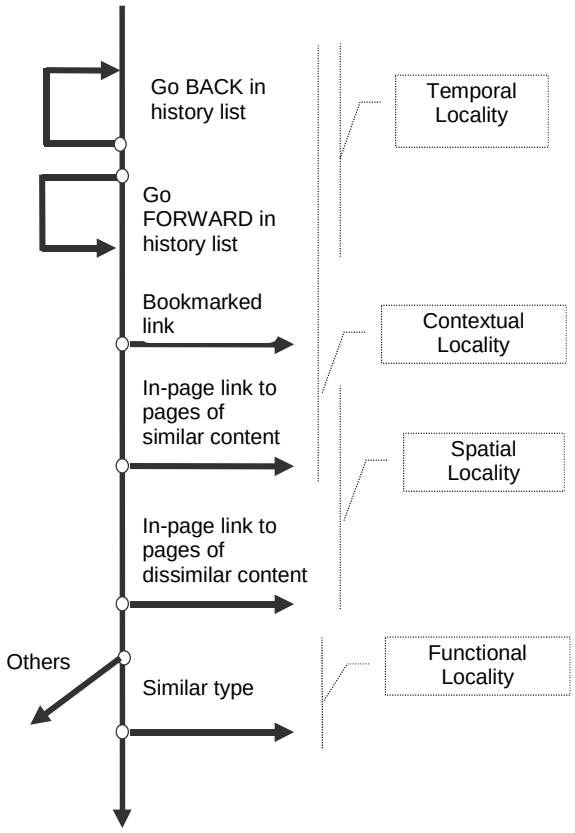

Figure 3 Web traversal

A. Temporal Locality

Objects accessed recently are deemed most likely to be accessed again in the near future. In a web cache, this locality is made possible by traversals via the browsers' "Back/Forward" buttons and the history list. This enables easy access to recently accessed objects. Temporal locality will be indicated by the existence of predictors such as "time since last access of an object" and "the number of times an object in accessed in a past window".

B. Spatial Locality

In a web context, objects are scattered all across the Web and it is unreasonable to expect spatial locality to exist in a physical sense. Even for accesses to a single site, these objects can be residing on different file servers on a network. We redefine spatial locality as objects being logically close to each other. Web objects that can be accessed directly via links on the current page are deemed more likely to be accessed next. Spatial locality is present if an object is requested as a primary object within a short time after it presents itself as a secondary object.

C. Functional Locality

We introduce the term functional locality to refer to the tendency to access objects performing similar functions. Banners, backgrounds, icons and images used by a site have a tendency to be reused throughout the site. Smith had proposed the use of separate instruction and data caches [2], and RISC machines have successfully employed such a strategy [16]. Such a design is based on the observation that instructions and data perform separate functions. In our terminology, these caches are believed to exhibit functional locality. Predictors include the type and size of an object. For example, "gif" and "jpg" both represent image files which are functionally similar.

D. Topical (Contextual) Locality

We also introduce the term topical (contextual) locality to refer to the tendency to access objects of the same topic or context. Topical locality is difficult to model in a traditional cache as memory objects are simply physical addresses. In this paper, we will limit our attention to keywords given in the title of an HTML document. We capture a user's attention span on a certain topic by tracking the appearance of such keywords. A comprehensive content determination will inevitably require complex linguistic parsing and dictionaries. In addition, the topical information is also indicated by the complexity of a page. This is represented by the number of links and images on the page.

Another source of predictors exists in a client's bookmark list. They give hints to both contextual and spatial locality. However, browser instrumentation is time-consuming and raises privacy concerns. As a result, we did not pursue this avenue.

V. The Logistic Regression Model

Having determined the predictors needed for our study, we now present a tool capable of determining the interactions and effects of these predictors.

The Logistic Regression (LR) model has been widely used by the medical community to model the risk (probability) of disease development (event) as a function of the risk factors [17]. For example, if high cholesterol levels are found to be predictive of heart diseases in a sample population, the same is assumed of the general population. Our goal is to express the outcome of the dependent variable Y, in terms of its predictors, $\mathbf{X} = (1, X_1,..., X_k)$ and their respective coefficients $(\beta_0,\beta_1,...,\beta_k)$.

The LR probability is given by:

$$P_{LR} = P(Y=g \mid 1, X_1,..., X_k) = \frac{1}{1+\exp(-z)}$$

$$\text{where } z = \sum_{j=0}^{k} \beta_j X_j \quad -\infty < \mathbf{z} < +\infty$$

Given a set of observed data, the coefficients can be estimated by a suitable method (learning phase). Once the coefficients are determined, the LR probability can be calculated for other objects (prediction phase).

In the simple case, Y takes on the value one or zero, according to whether or not a specific event occurs during the defined study period (time t_0 to t). In our context, the event of interest is whether a web object is re-accessed at least once, in the next W_F accesses. The lifespan of an object is inversely proportional to its re-access probability.

Let Y_i be the binary outcome (1 for an event, 0 for a nonevent) observed on the i-th access. Let $\mathbf{X}_i$ be the associated vector of predictors. Assuming $Y_1,Y_2,...,Y_m$ to be independent, the joint probability of the sequence $Y_1,Y_2,...,Y_m$ is proportional to the product

$$\prod_{i=1}^{m} P(Y=Y_i|\mathbf{X_i})$$

This is the likelihood equation. The optimal coefficients β*'s* are found by maximizing the likelihood equation. If $\beta=0$, the predictor has no influence. If $\beta>0$, it increases the probability of the event and vice versa. It should be noted that it is the entire product ($\beta\mathbf{X}$) that affects the outcome. Iterative algorithms, such as Newton-Raphson, are usually employed to perform the maximization. The p-value is commonly used to measure the strength of evidence that the estimates of β are statistically significantly different from zero [9]. In other words, if a predictor does not affect the probability of re-access, its corresponding p-value will indicate non-significance. For our purposes, we considered any predictors with p-value ≤ 0.1 to be significant.

VI. Implementation

We have selected the following predictors to represent the various localities:

$$Y = \begin{cases} 1 & \text{object re-accessed in given forward-looking window, } W_F \\ 0 & \text{otherwise} \end{cases}$$

Temporal Locality

X_1 = SINCE, time since last access in a backward-looking window, W_B.

X_2 = BHITS, number of hits in a backward-looking window, W_B

Functional Locality
X_3 = SIZE, size of object.
X_4 = TYPE, type of object (categorial variable)
1: HTML & text files e.g. html, txt
2: Image files e.g. jpeg, gif, jpg.
3: All others

Topical/Contextual Locality
X_5 = NUM_LINKS, number of hypertext links (secondary objects) on object.
X_6 = NUM_IMAGES, number of images on object
X_7= NUM_KEYWORDS, number of keywords (in title or description) that match a top 10 list.

Spatial Locality
Spatial locality is confirmed by determining if significant predictors exists in secondary objects. This implies that a secondary object is accessed as a primary object within a given forward-looking window. More details of dealing with secondary objects are given in the next section.

A typical trace contains the following information:

- IP address of requesting host (sanitized)
- time of request
- the requested Uniform Resource Locator (URL)
- bytes in reply (file size)
- reply status.

(Figure 4) shows our experimental setup. To ensure that our algorithm is truly adaptive across different sites, we have used trace data from three web servers (BIO, CAES, WHY) and three web proxies (CAEP, SJP, LJP) for our experiments (Figure 6). Client traces were not used for privacy reasons.

A. Extracting Predictors

We preprocessed the traces to remove badly formed lines and extraneous characters. Some servers have caching enabled. Subsequent access of the object is served directly from the cache. These files contain no size information, and we substitute it with the last known size from its trace history. For a CGI-query represented by existence of "?", we consider the part just before "?" as the object (it is most likely to be a file containing a query form). To ease comparison of secondary objects, relative paths are expanded to the absolute path.

B. Primary Objects

The event of re-access (Y) is set to 1 if there is at least one re-access within a forward window, W_F. The trace log gives us the size (SIZE) and type (TYPE) of an object. A moving-window of a history of W_B accesses keeps track of back hits (BHITS) and time since last accessed (SINCE).

We assumed that only HTML pages possess contextual predictors. These are obtained by downloading (playing back) the actual web page, and parsing through the HTML page. The number of links (NUM_LINKS) and number of images (NUM_IMAGES) are obtained by counting the number of <HREF> and <IMG> tags respectively.

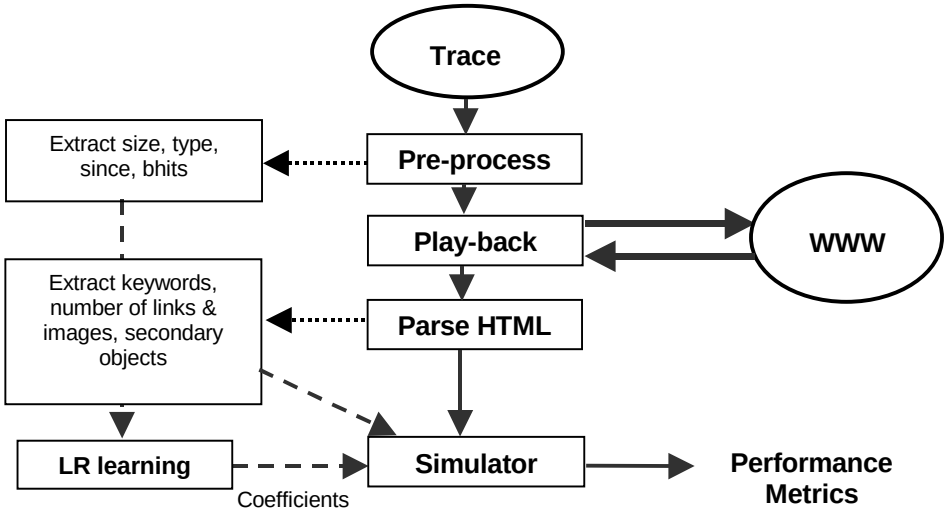

Figure 4 Experimental setup

The number of keywords (NUM_KEYWORDS) associated with an object requires more explanation. We obtain keywords from the title of an object. A first pass removes all stopwords from the title. Stopwords are those words which are judged to be so common in an online system that they have little information value [18] (Figure 5). The remaining words are considered as keywords. The top ten most frequently occurring keywords are noted. A second pass compares the keywords in the title and determines how many keywords are from the top ten list. The test is for words containing the given keywords. As such, plural forms of a word are considered the same, so are extensions of a given word (example, "computer" and "computerized" matches the keyword "computer"). Two passes are needed during the learning phase. Once the top ten list is determined, the prediction phase requires only one pass to assign NUM_KEYWORDS.

C. Secondary Objects

Secondary objects are potential candidates for pre-fetching. Only objects that a user must deliberately invoke for access, (i.e. <HREF> tags) are considered as secondary objects. In this manner, we take a more stringent approach then other researchers [19] where embedded objects (such as images) are also considered to be pre-fetch candidates.

Only primary HTML pages are considered to contain secondary objects. The event of re-access (Y), of a secondary object, is set to 1 if there is at least one re-access of it as a *primary* object, in W_F. The trace log gives us the type (TYPE) of an object. The size of a secondary object must be obtained via a header query of an object's content length. BHITS gives the number of times the secondary object was accessed as a primary object. SINCE is the time since it was last accessed as a primary object. NUM_LINKS and NUM_IMAGES of a secondary object, are the count of number of peer hypertext links and images that are on the HTML page. The keywords of a secondary object are extracted from the verbose description of the hypertext link (the "clickable" portion in a HTML page). The actual title of a secondary object can also be obtained via the optional "TITLE" tag in a link description. However, associated title tags were extremely rare in our trace. We therefore used the description of the link as an estimation of the subject matter.

A	a, about, above, across, adj, after, afterwards, again, against, albeit, all, almost, alone, along, already, also, although, always, among, amongst, an, and, another, any, anyhow, anyone, anything, anywhere, are, around, as, at
B	be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, below, beside, besides, between, beyond, both, but, by
C	can, cannot, co, could

Figure 5 Sample stopwords

Trace	Object	Coefficients associated with predictors
BIO	Primary	0.92 x BHITS -5.0e-6 x SINCE
	Secondary	-0.00004 x SIZE (-, -1.26, -1.26) x TYPE 0.02 x NUM_LINKS -0.02 x NUM_IMAGES 0.35 x BHITS -0.1e-5e-6 x SINCE
CAES	Primary	0.37 x BHITS -7.8e-7 x SINCE
	Secondary	7.1e-6 x SIZE 0.82 x BHITS -1.1e-6 x SINCE
WHY	Primary	-5.0e-5 x SIZE 2.02 x BHITS -2.4e-6 x SINCE
	Secondary	1.05e-4 x SIZE -0.073 x NUM_LINKS -0.037 x NUM_IMAGES -1.2 x BHITS -1.7e-6 x SINCE
CAEP	Primary	2.0e-5 x SIZE (1.12,-, 1.12) x TYPE 0.07 x BHITS -1.6e-6 x SINCE
	Secondary	1.2e-4 x SIZE (-,1.9, 1.9) x TYPE -0.02 x NUM_IMAGES
SJP	Primary	(1.17,-,1.17) x TYPE 0.011 x NUM_IMAGES 0.63 x BHITS -1.5e-3 x SINCE
	Secondary	-0.03 x NUM_LINKS -0.02 x NUM_IMAGES -7.5e-9 x SINCE
LJP	Primary	(-,-0.67,-0.67) x TYPE 0.45 x BHITS -1.5e-9 x SINCE
	Secondary	9.1e-3 x NUM_LINKS 1.5 x NUM_KEYWORDS

Figure 6 Significant predictors of re-access

The major drawback in our strategy is in its added complexity. However, the LR model is a relatively inexpensive process. The learning phase (which incurs the highest cost) took between 1.3 to 1.5 seconds of processing when done using a commercial package, SAS, on a Pentium 133Mhz system. Customized code will allow for even faster processing times. With web download times in the range of seconds, we believe that additional processing will not be a bottleneck. Our caching algorithm can theoretically be implemented as two background processes, continuously learning and

predicting. The cache strategy itself (prediction phase) requires a minimal overhead of computing the LR probability and sorting. For simplicity, we performed only one pass of learning and prediction. We based our learning on the number of accesses, instead of time, to circumvent the problem of possible down time on the servers. Furthermore, we ensured that learning is based on an equal number of samples for each trace. We had also specified our windows, W_F and W_B, in terms of accesses. We arbitrarily chose the following values for our experiments: W_F = 100, W_B = 100. These numbers were chosen to be the minimal required to give us enough events for usable statistical analysis. A rule of thumb used by biostatisticians is to have at least ten events for every predictor in the model. Obviously, with larger windows, a more accurate model can be achieved. We want to balance this against resources required to keep a longer history log. N_L=1000 traces were submitted as the learning data to obtain the ML estimates of the coefficients. Our prediction trace is set to N_P = 10,000. The predictors for the six traces are given in (Figure 6).

We made several important observations:

1. Multiple dimensions of locality exist. Different sites exhibit different types of locality.
2. All primary objects show strong temporal & spatial locality.
3. Secondary objects have less temporal locality but strong contextual locality. In other words, users are going to secondary objects with similar context, but not necessarily the same object.

Topical locality is not as pervalent as expected. This is probably due to the limitation of using only title keywords for context. Temporal locality is strongly evident for web servers. However, it is not so for secondary objects on proxy servers. Only topical and functional locality are evident. We interpret that a group of users served by a proxy is likely to be demographically similar and interested in similar topics. However, since a proxy server process requests of many users, temporal locality is not evident in a short window.

VII. APPLYING TO WEB CACHING

We have confirmed the existence of multi-dimensional locality in web accesses. We are also interested to know if lifespans, as predicted by the LR model, can be used to improve caching performance. We first determine if the prediction of lifespan is accurate. We do so by correlating the actual lifepsans obtained from traces to lifespans predicted by the LR model.

Policy	Criteria
LSIZE	Evict object with largest weighted-lifespan. Admit only if smaller. Perfect future knowledge.
LIFESPAN	Evict object with the largest lifespan. Admit only if smaller. Perfect future knowledge.
LR-LSIZE	Evict object with the largest predicted weighted-lifespan. Admit only if smaller
LR-LIFESPAN	Evict object with the largest predicted lifespan. Admit only if smaller
LRU	Least Recently Used. Evict object with the largest time since last access
LFU	Least Frequently Used. Evict object with the smallest number of accesses
FIFO	First In First Out. Evict object with the earliest time of entry to cache
SIZE	Evict object with the largest size.
Random	Randomly chosen

Figure 7 Cache strategies

A positive correlation would indicate that large values of actual lifespans are linked to large values of predicted lifespans. Since cache strategies are implemented based on relative ranks of lifespans (i.e. re-access probabilities), a positive correlation alone is sufficient for our purposes. We found that the predicted lifespans correlate positively (between 0.42 – 0.55) for all six traces. These values are also statistically significant (p-value < 0.05).

For our cache experiments, we determined the minimum cache size needed for emulating an infinite cache for each of the traces. We ran simulation on cache sizes that were 1%, 5%, 10% and 20% relative to the "infinite" cache size. Hence, we ensured that each trace had ample chance to "warm" the cache and comparisons are standardized across the different traces. We had also determined the hit ratios if we had "infinite" cache, and present results relative to this ideal.

Our Logistic Regression-based algorithms are called LR-LIFESPAN and LR-LSIZE. On the arrival of a new object, the object with the largest lifespan (or weighted lifespan) will be evicted. If the new object has a larger lifespans (or weighted lifespans) than existing objects, it will not be cached at all. Any object deemed to have an infinite lifespan will not be admitted into cache.

We compared the performance of our policies (LR-LIFESPAN, LR-LSIZE) to the more common policies — SIZE, LRU, FIFO, LFU, and random (Figure 7). We did not use the variations proposed by other researchers (e.g. LFU-SIZE, LRU-SIZE, etc) as there can be

potentially be many combinations. We are more interested in the LR algorithms as a predictive tool. As such we are better served by comparing with the generic versions of each algorithm. LRU, FIFO and LFU essentially utilize only one predictor, while LR-based algorithms use as many significant predictors that there are. We have also included the performance of algorithms with perfect future knowledge (LSIZE, LIFESPAN) to see the effects of perfect prediction.

Our results showed LR-LSIZE have the best performance optimizing object hit ratios (Figure 8). If our purpose is to improve byte hit ratio (Figure 9), LR-LIFESPAN should be used. Due to its selective nature, LR-LSIZE also have the added advantage of having very small turnover ratios (results not reported here). A small turnover ratio is useful in reducing the need for constant cache maintenance. The differences in performance are also more obvious for smaller caches.

To provide some perspective to what the improvements mean, we compare some examples of realistic timing measures of LR-LSIZE to that of FIFO (used by commercial caches).

A user's perception of response time is mainly governed by the time to arrival of the first byte of data. Ignoring delays incurred by hits,

Expected first-byte arrival = $(1-h) * \delta_{avg}$

h = hit ratio

δ_{avg} = average delay from non-cache location

Experiments showed average delays over UW's campus backbone to be 34 ms. Object hit ratios incurred by LR-LSIZE are up to 40% better than those incurred by FIFO. For example, in one situation, LR-LSIZE has a hit ratio of 0.46, while FIFO has a hit ratio of 0.23. This translates to expected first-byte arrival of 18ms and 29 ms respectively. The differences in delays are obviously not noticeable to a human. Bickford reported that 200ms forms the mental boundary between events that seem to happen together [20]. However, the difference will become significant with larger delays, as in the case of WAN or modem connections. Average modem delays as small as 1s [5] will result in expected propagation delays of 540ms for LR-LSIZE, and 770ms for FIFO. This difference will be perceivable to the user.

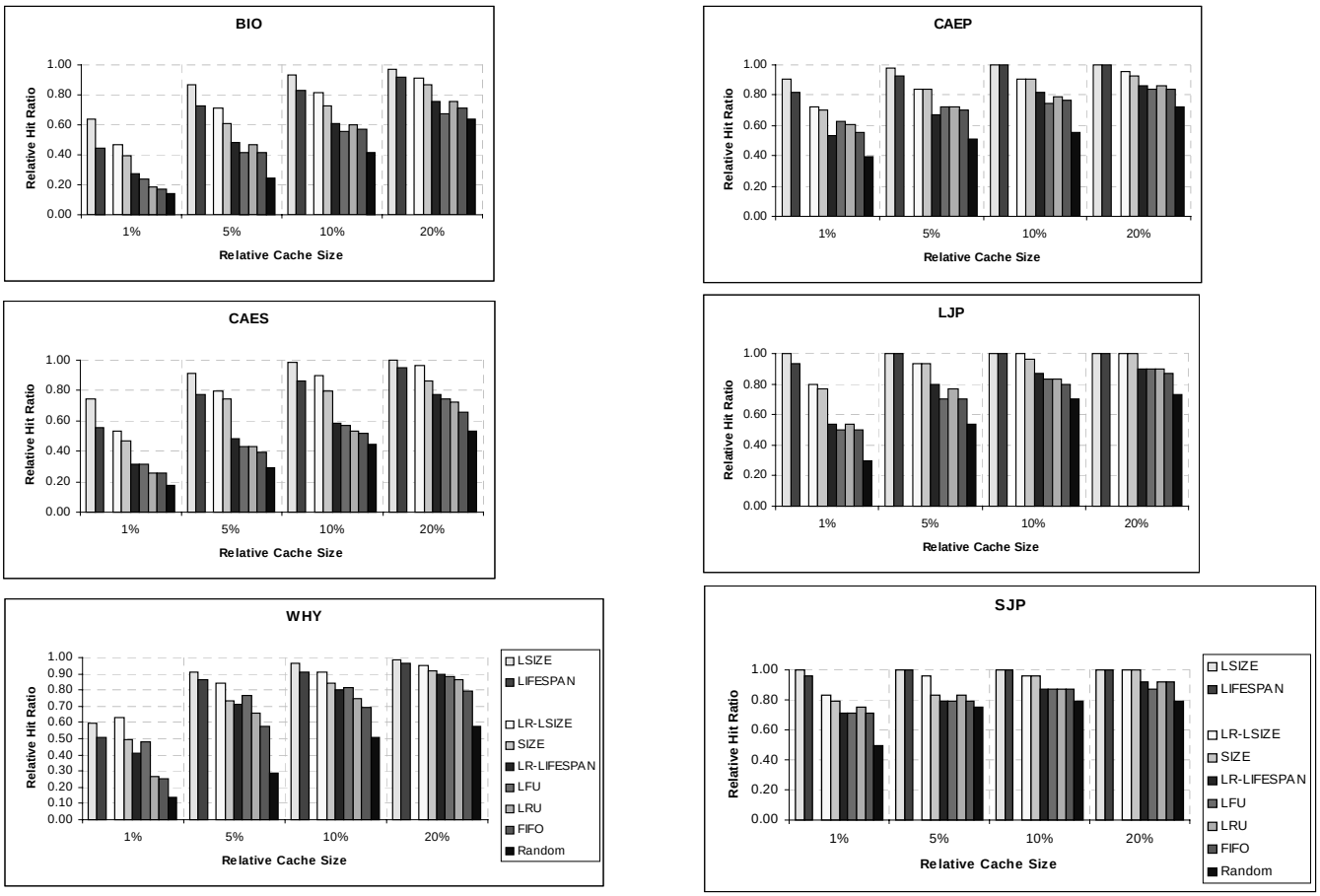

Figure 8 Object hit ratios

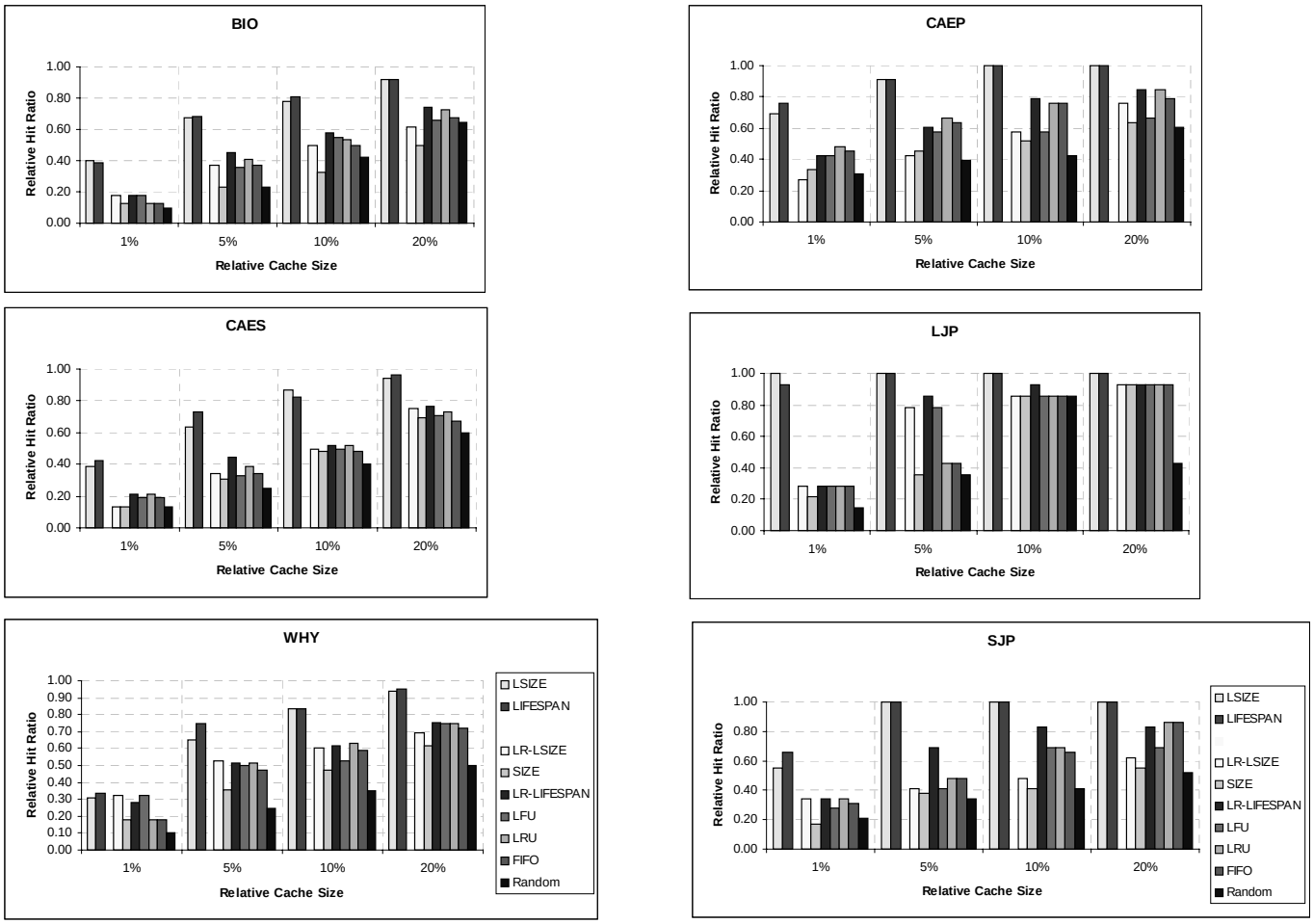

Figure 9 Byte hit ratios

VIII. FUTURE WORK

We did not address the issue of pre-fetching in this paper. Nevertheless, the framework we have developed applies equally well to pre-fetching. The existence of spatial locality in all instances strongly suggests that pre-fetching can be exploited. Commercial products, such as NetAccelerator and PeakJet, pre-fetches as many hypertext links as possible [21]. Initial pre-fetch experiments using LR prediction showed that we can achieve hit ratios similar to brute-force pre-fetches with far less traffic. Researchers in pre-fetching usually construct dependency graphs that notes the number of times a file follows another file. An arbitrary threshold value is set, above which pre-fetch will occur [19]. These methods associate *specific* objects (based on URLs) with one another. That is to say, if an object changes its name or location, the dependency constructed earlier would be invalid. In addition, a cache must have first seen a direct relationship between two objects in order to construct such a dependency graph. The strength of a LR-based pre-fetch strategy is that we can predict accesses of a secondary object based on its set of predictors, not on a previously established link, or a static name.

Finally, we hope to further improve on the extraction of topical locality. The trend towards associating more meta-data and descriptions with web objects, will provide us with better predictors to work with. This will ultimately enables us to work on what we hope to introduce as *content-aware caching*.

IX. CONCLUSION

We have identified the existence of multi-dimensional locality in web accesses. Design of cache strategies must account for all of these localities to be fully effective. The localities and predictors presented in this paper is by no means exhaustive. What we have achieved here is show the applicability of a simple LR model in characterizing re-access probability. Such a model can also prove to be a useful tool for anlaysing

site designs. Futhermore, the availability of eventful data in web re-accesses is especially suited towards such statstical methods as the LR model. In addition, we have provided a unified framework upon which all three actions of a cache can be built. Finally, the existence of different predictors of web access at different sites suggests that a cache algorithm must be adaptive.

We have investigated but a small area in web caching. Specifically, we have made decisions based only on what a single cache sees. The effects of other caches are unknown. Our hope is that by understanding single caches, we lay the foundation for understanding the more complex relationships among cooperating caches.

ACKNOWLEDGMENT

We would like to thank Mike Gerdts, Darrell Schulte, Jinah Yun-Mitchell and Duane Wessels, for providing us with the web traces used in this study. We would also like to thank David Stern for converting this document to the proper format.

REFERENCES

[1] D. Strom, "The caching question," *Internet World*, Sept 1999.
[2] A.J. Smith, "Cache memories," *Computing Surveys*, vol. 14(3), 1982, pp. 473-530.
[3] L.A. Belady, "A study of replacement algorithms for a virtual-storage computer," *IBM Systems Journal*, vol. 5(2), 1966, pp. 78-101.
[4] G.M. Voelker, T.K. Anderson, M.J. Feeley, J.S. Chase, A.R. Karlin, and H.M. Levy, "Implementing cooperative prefetching and caching in a globally-managed memory system," *Proc. of the 1998 ACM SIGMETRICS Conf. on Performance Measuring, Modeling and Evaluation*, Jun 1998.
[5] S. Hosseini, "Investigation of generalized caching," Ph.D. thesis, Washington University, 1997.
[6] M. Abrams, C.R. Standridge, G. Abdulla, S. Williams and E.A. Fox, "Caching Proxies: Limitations and Potentials," Virginia Tech Report, TR-95-12, 1995.
[7] S. Williams, M. Abrams, C.R. Standridge, G. Abdulla and E.A. Fox, "Removal Policies in Network Caches for WWW Documents," *Proc. of ACM SIGCOMM 96*, Stanford, Aug 1996.
[8] I. Tatarinov, A. Rousskov, V. Soloviev, "Static caching in web servers," *Proc. of the 6th IEEE Intl. Conf. on Computer Communications & Networks*, Las Vegas, 1997.
[9] R.P. Wooster, "Optimizing response time, rather than hit rates of WWW proxy caches," Masters Thesis, 1996.
[10] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "Enhancing the web's infrastructure: From caching to replication," *IEEE Internet Computing*, April 1997, pp. 18-27.
[11] J.E. Pitkow and M.M. Recker, "A simple yet robust caching algorithm based on dynamic access patterns," *Proc. 2nd Intl. WWW Conf.*., Chicago, 1994.
[12] Lorenzetti, P., Rizzo, L., and Vicisano, L, "Replacement policies for a proxy cache," 1997; available at http://www.iet.unipi.it/~luigi/caching.ps.gz.
[13] M.F. Arlitt and C.L. Williamson, "Web server workload characterization: The search for invariants," *ACM SIGMETRICS Conference*, ACM Press, Philadelphia, 1996.
[14] A.P. Foong, "An adaptive cache strategy for the WWW," Ph.D. thesis, University of Wisconsin-Madison, Dec 1999.
[15] A.P. Foong, Y. Hu and D.M. Heisey, "Essence of an effective web caching algorithm," unpublished.
[16] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*. McGraw-Hill, Inc., New York, 1993.
[17] D.W. Hosmer and S. Lemeshow, *Applied Logistic Regression*. John Wiley & Sons, New York, 1989.
[18] "Query stopword list for THOMAS," THOMAS: Legislative information on the Internet 10-12-1998; available at http://rs9.loc.gov/home/stopwords.html.
[19] P. Bickford, "Human interface online: Worth the wait ?," 1997; http://developer.netscape.com/viewsource/bickford_wait.htm.
[20] B.L. Georgia, "More web, less wait," *BYTE*, July 1998.
[21] V.N. Padmanabhan and J.C. Mogul, "Using predictive prefetching to improve WWW latency," *Computer Communications Review*, vol. 26, Jul 1996, pp. 22-36.